LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

# CS4001NI Programming

## 30% Individual Coursework

## 2023-24 Autumn

**Student Name: MD INJAMAMUL HAQUE**

**London Met ID: 23049321**

**College ID: NP01CP4A230260**

**Group:** **C10**

**Assignment Due Date: Tuesday, December 26, 2023**

**Assignment Submission Date: Friday, January 26, 2024**

## Table of Contents

## 1. Introduction:

This is the first coursework given by Islington college to us. The main aim of this coursework assignment was to implement a real-world scenario and using OOP concept of Java which includes creating a parent class as a Teacher, together with it's subclasses to Lecturer and Tutor respectively. In this assignment the Java program is to be run in Bluej with proper order and formatting according to the provided questions. In this coursework we have to use various methods such as get, set, display and other to execute the code.

## 1.2.Bluej:

BlueJ is an integrated development environment for the Java programming language, developed mainly for educational purposes. It runs with the help of JDK (Java Development Kit).Blue was an integrated system with its own programming language and environment. BlueJ implements the Blue environment design for the Java programming language. (Ali, September 2018)

## 1.3.Java:

Java is a popular programming language used in app development, desktop computing, and gaming. It was created by James Gosling, Mike Sheridan, and Patrick Naughton of Sun Microsystems, later acquired by Oracle. When Java was introduced in 1995, it was marketed for smart TVs; ironically, the technology was too advanced for cable TV at the time. Java soon became an industry standard for internet programming due to its ability to run on any platform.

According to the TIOBE programming language popularity index, Java has consistently ranked as a top 10 programming language from mid-2015 to 2021. In fact, as of September 2021, Java is the #1 most popular programming language. This could be why Java remains the primary language used on the AP Computer Science exam.

Knowledge of Java and its applications will undoubtedly help students in their computer science education. Let’s dive deeper into the characteristics and syntax of Java, then explore professional opportunities available to Java developers. (Baird, Ovtober 21, 2021)

## 1.4.Draw.io:

Nowadays, data has become an essential asset for both the individual and business. However, it is impossible to understand the data for average people. Therefore, data visualization becomes an important skill. The diagramming applications can help people to turn data into graphics and charts for presentations. This article will review one of the popular diagramming tools, Draw.io. So that you can get the key information, such as features, benefits, and more, before using it.Draw.io is a fully featured diagramming application designed to serve enterprises, small businesses, agencies, and individuals. As a free online platform, it makes diagram drawing portable. Therefore, it is an excellent option for both professionals and average people to create visual content for free. (Mae, November 18, 2022)

## 1.5.Ms-Word:

Microsoft Word is a word processing program that was first developed by Microsoft in 1983. Since that time, Microsoft has released an abundance of updated versions, each offering more features and incorporating better technology than the one before it. The most current web-based version of Microsoft Word is Microsoft 365, but the software version of Microsoft Office 2019 includes Word 2019.Microsoft Word is included in all of the Microsoft 365 application suites. The most basic (and least expensive) suites also include Microsoft PowerPoint and Microsoft Excel. Additional suites exist and include other Office programs, such as Microsoft Outlook and Skype for Business. With Microsoft Word you can choose from a variety of preconfigured styles and designs, which provides an easy way to format long documents with just a single click. You can also insert pictures and videos from your computer and the internet, draw shapes, and create an insert all kinds of charts. (Ballew, June 12,2021)

## 2.Class Diagram:

The class diagram is one of the types of UML diagrams which is used to represent the static diagram by mapping the structure of the systems using classes, attributes, relations, and operations between the various objects. A class diagram has various classes; each has three-part; the first partition contains a Class name which is the name of the class or entity which is participated in the activity, the Second partition contains class attributes that show the various properties of the class, the third partition contains class operations which shows various operations performed by the class, relationships shows the relation between two classes.In a class diagram, it is necessary that there exists a relationship between the classes. Unfortunately, the similarity of various relationships often makes it difficult to understand them. (Pedamkar, March 13,2023)

## 2.1. Class Diagram of Teacher:

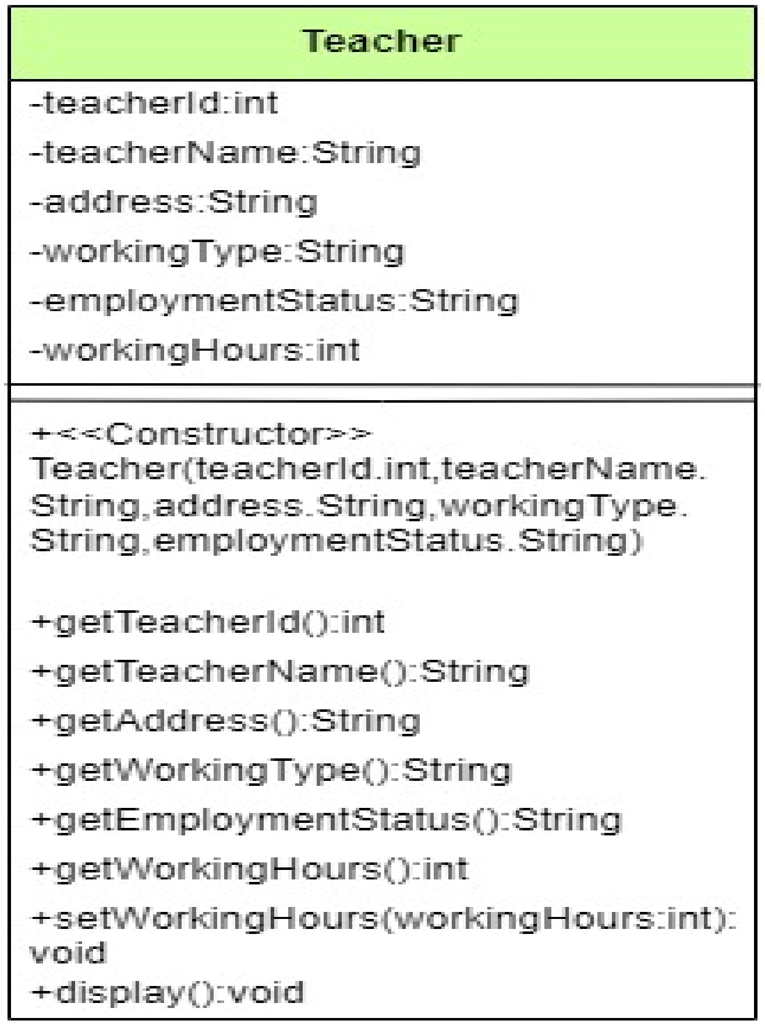


*Figure 0-1:Class diagram of Teacher*

## 2.2. Class diagram of Lecturer:

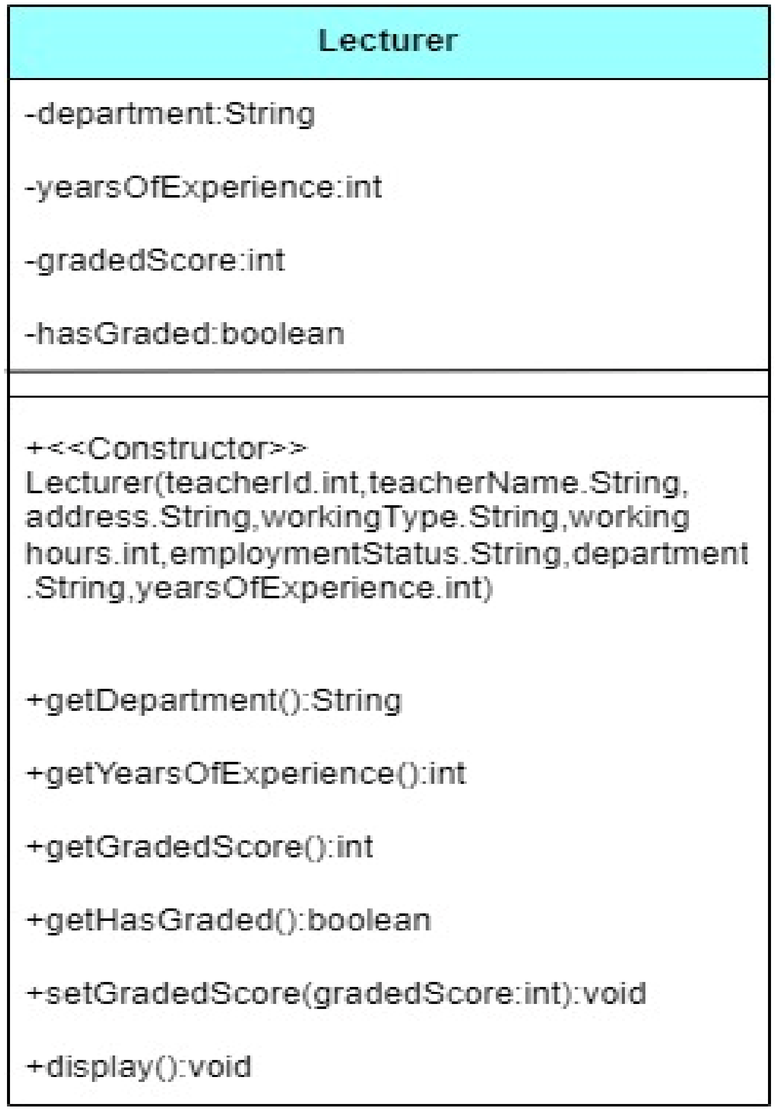

Lecturer
-department:String
-yearsOfExperience:int
-gradedScore:int
-hasGraded:boolean
+<<Constructor>> Lecturer(teacherId.int,teacherName.String, address.String,workingType.String,working hours.int,employmentStatus.String,department .String,yearsOfExperience.int)
+getDepartment():String
+getYearsOfExperience():int
+getGradedScore():int
+getHasGraded():boolean
+setGradedScore(gradedScore:int):void
+display():void

*Figure 0-2:Class diagram of Lecturer*

## 2.3:Class diagram of Tutor:

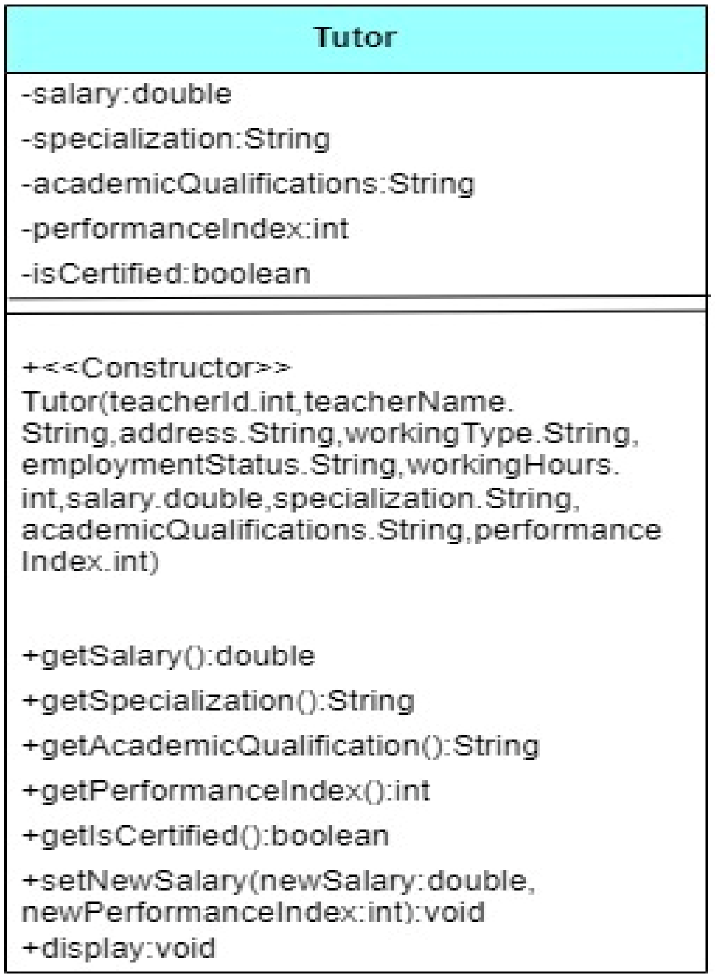

Tutor
-salary:double
-specialization:String
-academicQualifications:String
-performanceIndex:int
-isCertified:boolean
+<<Constructor>>
Tutor(teacherId.int,teacherName.
String,address.String,workingType.String,
employmentStatus.String,workingHours.
int,salary.double,specialization.String,
academicQualifications.String,performance
Index.int)
+getSalary():double
+getSpecialization():String
+getAcademicQualification():String
+getPerformanceIndex():int
+getIsCertified():boolean
+setNewSalary(newSalary:double,
newPerformanceIndex:int):void
+display:void

*Figure 0-3: Class diagram of Tutor*

## 2.4.Diagram of Relationship between Teacher,Lecturer and Tutor:

In the relationship between these three classes Teacher is a parent class of both child class/sub class of Lecturer and Tutor.

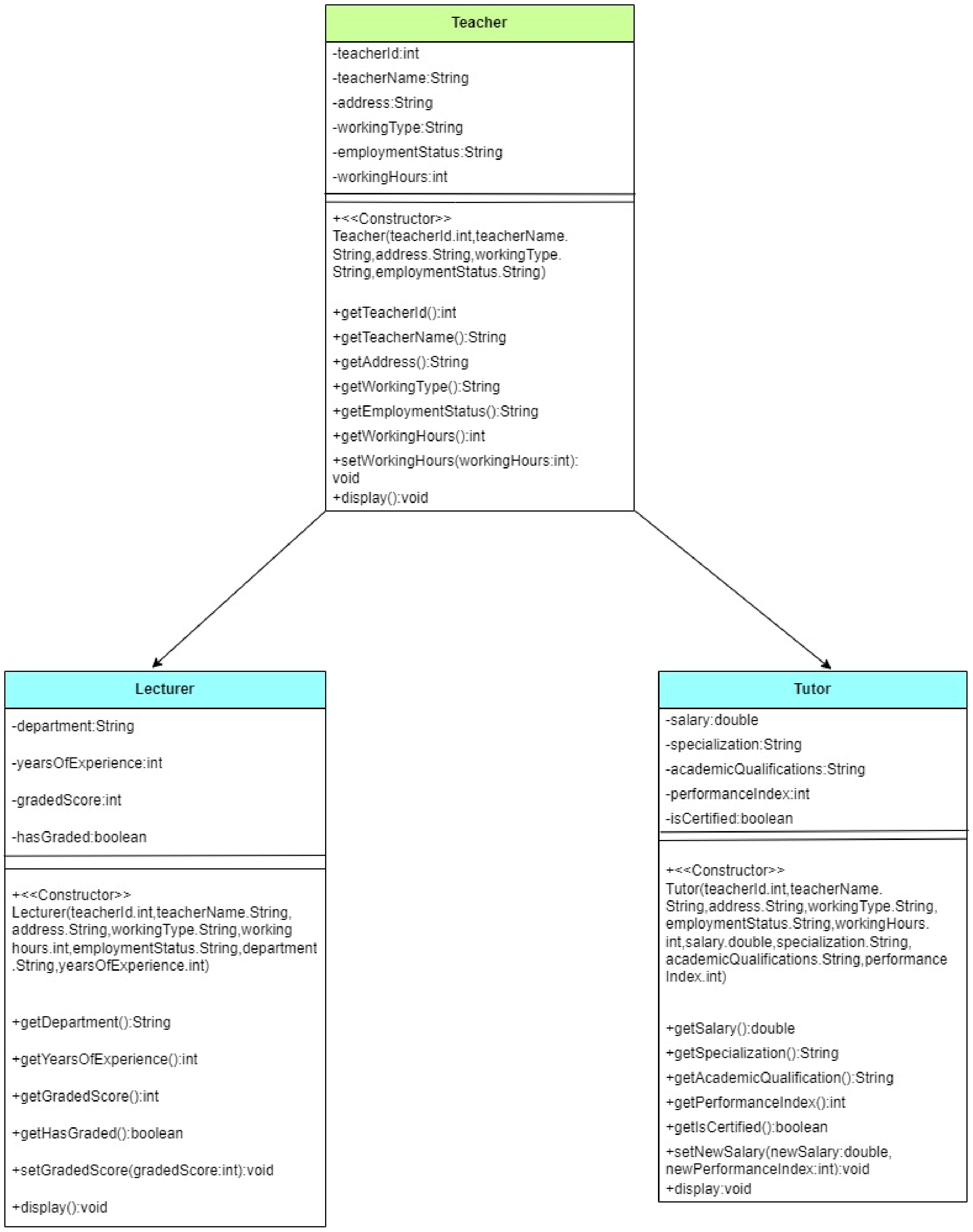

Teacher
-teacherId:int
-teacherName:String
-address:String
-workingType:String
-employmentStatus:String
-workingHours:int
+<<Constructor>> Teacher(teacherId.int,teacherName. String,address.String,workingType. String,employmentStatus.String)
+getTeacherId():int
+getTeacherName():String
+getAddress():String
+getWorkingType():String
+getEmploymentStatus():String
+getWorkingHours():int
+setWorkingHours(workingHours:int): void
+display():void
Lecturer
-department:String
-yearsOfExperience:int
-gradedScore:int
-hasGraded:boolean
+<<Constructor>> Lecturer(teacherId.int,teacherName.String, address.String,workingType.String,working hours.int,employmentStatus.String,department .String,yearsOfExperience.int)
+getDepartment():String
+getYearsOfExperience():int
+getGradedScore():int
+getHasGraded():boolean
+setGradedScore(gradedScore:int):void
+display():void
Tutor
-salary:double
-specialization:String
-academicQualifications:String
-performanceIndex:int
-isCertified:boolean
+<<Constructor>> Tutor(teacherId.int,teacherName. String,address.String,workingType.String, employmentStatus.String,workingHours. int,salary.double,specialization.String, academicQualifications.String,performance Index.int)
+getSalary():double
+getSpecialization():String
+getAcademicQualification():String
+getPerformanceIndex():int
+getIsCertified():boolean
+setNewSalary(newSalary:double, newPerformanceIndex:int):void
+display:void

***Figure 0-4:Relationship between parent class and child class***

## 3.Pseudo codes:

Pseudo code is a way of representing an algorithm or process using a formalized notation that is easier to understand than standard programmatic code. Pseudo code is not executed by computers but is instead used as a guide for developers when implementing algorithms in software.

Pseudo code typically uses simple, concise language that closely resembles the structure of a programming language.

However, it is not constrained by the syntax or semantics of any particular programming language. As a result, pseudo code can be used to describe algorithms that can be implemented in any language.

Pseudo code is an invaluable tool for developers, as it enables them to clearly communicate their ideas before starting to write code. By taking the time to write pseudo code and algorithm, developers can save time and avoid errors when implementing the algorithm in software. (Obvii, October 7 2022)

### 3.1.Pseudo code of class Teacher:

```
CREATE a parent Class Teacher
DO
   DECLARE instance variable teacherId as int
   DECLARE instance variable teacherName as String
   DECLARE instance variable address as String
   DECLARE instance variable workingType as String
   DECLARE instance variable employmentStaus as String
   DECLARE instance variable workingHours as int


    CREATE Constructor Teacher(int teacherId, String teacherName, String
address, String workingType, String employmentStatus)
```

```
        ASSIGN this.teacherId to  teacherId
        ASSIGN this.teacherName to teacherName
        ASSIGN this.address to address
        ASSIGN this.workingType to workingType
        ASSIGN this.employmentStatus to employmentStatus


    CREATE an accessor method int getTeacherId() with return type string
DO
        RETURN teacherId
    END DO

    CREATE create an accessor method String getTeacherName() with return
type string
DO
        RETURN teacherName
    END DO

    CREATE an accessor method String getAddress() with return type string
DO
        RETURN address
    END DO

    CREATE an accessor method String getWorkingType() with return type string
DO
        RETURN workingType
    END DO

    CREATE an accessor method String getEmploymentStatus() with return type
string
DO
```

```
        RETURN employmentStatus
    END DO

    CREATE an accessor method int getWorkingHours() with return type string
DO
        RETURN workingHours
    END DO

    CREATE mutator method void setWorkingHours(int workingHours)
        ASSIGN this.workingHours to workingHours
    END

    CREATE Method void display()
        PRINT Teacher ID is calling from accessor method teacherId
        PRINT Teacher Name is calling from accessor method teacherName
        PRINT Address is calling from accessor method address
        PRINT Working Type is calling from accessor method workingType
        PRINT Employment Status is calling from accessor method
employmentStatus

        IF workingHours is less than or equal to 0
            PRINT "Working Hours: Not assigned please enter the working hour"
        ELSE
            PRINT Working Hours is calling from accessor method workingHours
        END IF
```

## 3.3.Pseudo code of class Lecturer:

```
        CREATE subclass Lecturer EXTENDS Teacher
  DECLARE instance variable department as string
  DECLARE instance variable yearsOfExperience as int
  DECLARE instance variable gradedScore as int
  DECLARE instance variable hasGraded as boolean


  CREATE Constructor Lecturer(int teacherId, String teacherName, String address,
String workingType, int workingHours, String employmentStatus, String department, int
yearsOfExperience)
    // Call superclass constructor
    CALL Super(teacherId, teacherName, address, workingType, employmentStatus)
    // Set working hours using superclass method
    CALL Super.setWorkingHours(workingHours)

    // Initialize additional attributes
    ASSIGN department to department
    ASSIGN yearsOfExperience to yearsOfExperience
    ASSIGN gradedScore to 0
    ASSIGN hasGraded to false
```

```
   CREATE an accessor method String getDepartment() with return type string
DO
      RETURN department
   END DO

   CREATE an accessor method int getYearsOfExperience() with return type int
DO
      RETURN yearsOfExperience
   END DO

   CREATE an accessor method int getGradedScore() with return type int
DO
      RETURN gradedScore
   END DO

   CREATE mutator method void setGradedScore(int gradedScore)
      ASSIGN this.gradedScore to gradedScore
   END

   CREATE accessor method boolean getHasGraded() with return type boolean
DO
```

```
        RETURN hasGraded
    END DO


    CREATE Method void gradeAssignment(int score, String department, int
yearsOfExperience)
        IF NOT hasGraded AND yearsOfExperience greater than or equal to  5 AND
department.equals(this.department)
            IF score is greater than or equal to  70 AND  less than or equal to  100
                ASSIGN gradedScore equal to  70
            ELSE IF score is greater than or equal to  60
                ASSIGN gradedScore equal to  60
            ELSE IF score is  greater than or equal to  50
                ASSIGN gradedScore equal to  50
            ELSE IF score  is greater than or equal to 40
                ASSIGN gradedScore equal to  40
            ELSE
                ASSIGN gradedScore equal to  0
            END IF


            ASSIGN hasGraded equal to  true
            ASSIGN this.gradedScore to Score


            PRINT is Assignment has been graded successfully
        ELSE
            PRINT is Assignment has not been graded yet
        END IF

    CREATE Method void display()
        CALL Super.display()
        PRINT Department calling from accessor method getdepartment
        PRINT Years Of Experience calling from accessor method getyearsOfExperience
        IF hasGraded
            PRINT Graded Score calling from getscore
        ELSE
            PRINT is Score has not been graded yet
        END IF
```

## 3.4.Pseudo code of class Tutor:

```
CREATE subclass Tutor EXTENDS Teacher
       DECLARE instance variable salary as double
       DECLARE instance variable specialization as string
       DECLARE instance variable academicQualification as string
       DECLARE instance variable performanceIndex as int
       DECLARE instance variable isCertified as boolean

    CREATE Constructor Tutor(int teacherId, String teacherName, String address, String
workingType, String employmentStatus, int workingHours, double salary, String
specialization, String academicQualifications, int performanceIndex)
        // Call superclass constructor
        CALL Super(teacherId, teacherName, address, workingType, employmentStatus)
        // Set working hours using superclass method
        CALL Super.setWorkingHours(workingHours)

        // Initialize additional attributes
        ASSIGN this.salary is salary
        ASSIGN this.specialization is specialization
        ASSIGN this.academicQualifications is academicQualifications
        ASSIGN this.performanceIndex is performanceIndex
        ASSIGN this.isCertified is false
    E

    CREATE an accessor method double getSalary() with return type double
DO
        RETURN salary
    END DO

    CREATE an accessor method String getSpecialization() with return type string
DO
        RETURN specialization
    END DO

    CREATE an accessor method String getAcademicQualifications() with return type
string
DO
        RETURN academicQualifications
    END DO

    CREATE an accessor method int getPerformanceIndex() with return type int
DO
        RETURN performanceIndex
    END DO
```

```
CREATE an accessor method boolean getIsCertified() with return type boolean
    RETURN isCertified
END

CREATE Method void setSalary(double newSalary, int newPerformanceIndex)
    IF newPerformanceIndex is less than 5 AND getWorkingHours() is greater than 20
        DECLARE double appraisal TO 0
        IF newPerformanceIndex less than equal to 5 AND newPerformanceIndex less than equal to 7
            ASSIGN appraisal to 0.05
        ELSE IF newPerformanceIndex less than or equal 8 AND newPerformanceIndex less than 9
            ASSIGN appraisal to 0.1
        ELSE IF newPerformanceIndex is equal to 10
            ASSIGN appraisal to 0.2
        END IF

        ASSIGN this.salary equal TO newSalary + (appraisal * newSalary)
        ASSIGN this.isCertified TO true
    ELSE
        PRINT is The salary cannot be approved since Tutor has not been certified
    END IF
    CREATE Method void removeTutor()
        IF NOTisCertified
        ASSIGN this.salary to 0
        ASSIGN this.specialization to null
        ASSIGN this.academicQualifications to null
        ASSIGN this.performanceIndex to 0
        ASSIGN this.isCertified to false
        ELSE
PRINT is Certified Tutor cannot be removed
END IF
```

```
    CREATE Method void displayTutorDetails()
        // Call superclass display method
        CALL Super.display()
        IF NOT isCertified
            PRINT Salary calling from accessor method getsalary
            PRINT Specialization calling from accessor method getspecialization
            PRINT Academic Qualifications calling from accessor method
getacademicQualifications
            PRINT Performance Index calling from accessor method getperformanceIndex
```

```
ELSE
    PRINT  is The Tutor has not been certified yet.
 END IF
```

## 4.Description methods of all classes:

There are several methods used in all three classes some of them are given below and the description of those all three classes are as follows:

### 4.1.Teacher:

Constructor:

The Teacher parent class is created using a constructor that accepts parameters to initialize its attributes. The provided arguments are used to set values for teacherId, teacherName, address, workingType, and employmentStatus in accordance with different datatype.

Getter Methods:

To access the attribute values, getter methods/accessor method are available for each attribute. These methods can be used by external code to retrieve the values.

Setter Method:

For the workingHours attribute, there is a setter method/mutator method called setworkingHours. This method allows external code to assign a value to the working hours.

Display Method:

The display method is responsible for showing the teacher's details, including teacherId, teacherName, address, workingType, employmentStatus, and workingHours (if it is greater than zero).Here, void displayTeacherDisplay is a method. If the workingHours attribute is not assigned (less than or equal to zero), a message is displayed indicating that working hours are not assigned.

## 4.2.Lecturer:

Constructors:

The attributes of the Lecturer subclass are initialized through a constructor that takes parameters. The constructor of the superclass (Teacher) is invoked using the super keyword. The workingHours attribute is also set using the setter method from the superclass. After that, the additional attributes of the Lecturer class (department, yearsOfExperience, gradedScore, and [hasGraded] are initialized.

Getter Methods:

To obtain the values of the additional attributes of the Lecturer class (department, yearsOfExperience, gradedScore, hasGraded), there are getter methods available. These getter methods/accessor allow external code to retrieve the values of these attributes.

Setter Method:

The gradedScore attribute has a setter method/mutator method (setgradedScore) that allows external code to assign a value to the graded score.

Method for Grading Assignments (gradeAssignment):

The gradeAssignment method assesses assignments based on the provided score, department, and years of experience. It checks certain conditions before assigning a grade. If the conditions are met, a grade is assigned to the gradedScore attribute and hasGraded is set to true.

Overridden display Method:

The display method is modified to include the additional attributes of the Lecturer class. It first calls the display's method of the superclass (Teacher) using super.display(). Then, it adds the display of department, yearsOfExperience, and, if graded, the gradedScore.

### 4.3.Tutor:

Constructors:

The class has a constructor that takes parameters to initialize the attributes of the Tutor object. It uses the super keyword to call the constructor of the superclass (Teacher). It also sets the workingHours attribute using the setter method from the superclass. The additional attributes of the Tutor class (salary, specialization, academicQualifications, performanceIndex, and isCertified) are then initialized.

Getter Methods:

There are getter methods for each additional attribute of the Tutor class (salary, specialization, academicQualifications, performanceIndex, isCertified), allowing external code to retrieve the values of these attributes.

Setter Method (setSalary):

This method sets the salary of the tutor based on the new salary and performance index. It includes conditions for salary approval based on the performance index and working hours. If the tutor is certified, the salary is adjusted with an appraisal, and the tutor is marked as certified.

Setter Method (removeTutor):

This method sets all attributes to default values and marks the tutor as not certified. It simulates removing a tutor if they are not certified.

Display Method (displayTutorDetails):

The displayTutorDetails method is used to display all details of the tutor, including the additional attributes. If the tutor is not certified, a message indicating that the tutor has not been certified yet is displayed.

These are the method description of Teacher, Lecturer and Tutor.

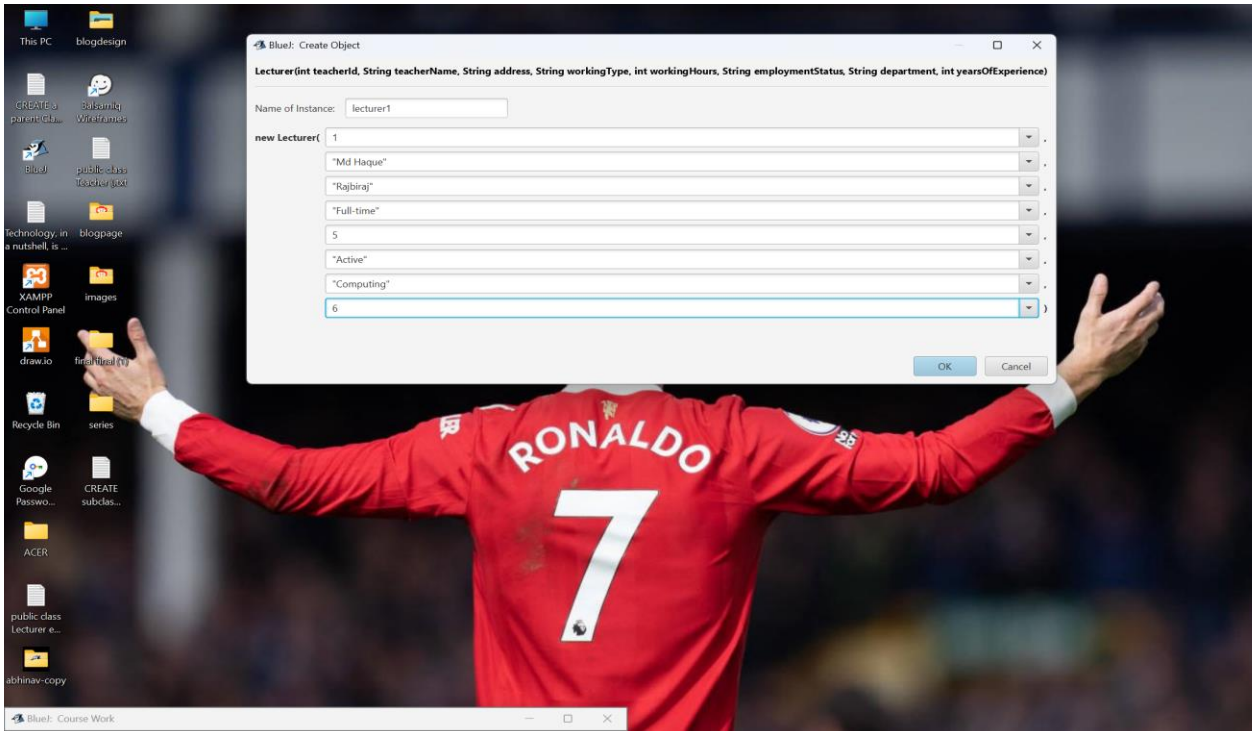
## 5.Testing :

### 5.1.Test 1 – To Inspect the Lecture class, grade the assignment, and re-inspect the Lecturer Class

| | |
|---|---|
| Test No: | 1 |
| Objective: | To inspect the Lecturer class, grade the assignment, and re-inspect the Lecturer class. |
| Action: | →The Lecturer is called with the following arguments:<br>teacherId=1<br>teacherName="Md Haque"<br>address="Rajbiraj"<br>workingType="Full-time"<br>workingHours=5<br>employmentStatus="Active"<br>department="Computing"<br>yearsOfExperience=6<br>→Inspection of the Lecturer class.<br>→void gradeAssignment(int score,String department, int yearsOfExperience)<br>int score=70<br>String department="Computing"<br>Int yearsOfExperience=6<br>→Re-inspection of the Lecturer class. |
| Expected Result: | Lecturer would assign for the gradedScore. |
| Actual Result: | The gradedScore was assigned. |
| Conclusion: | The test is successful. |

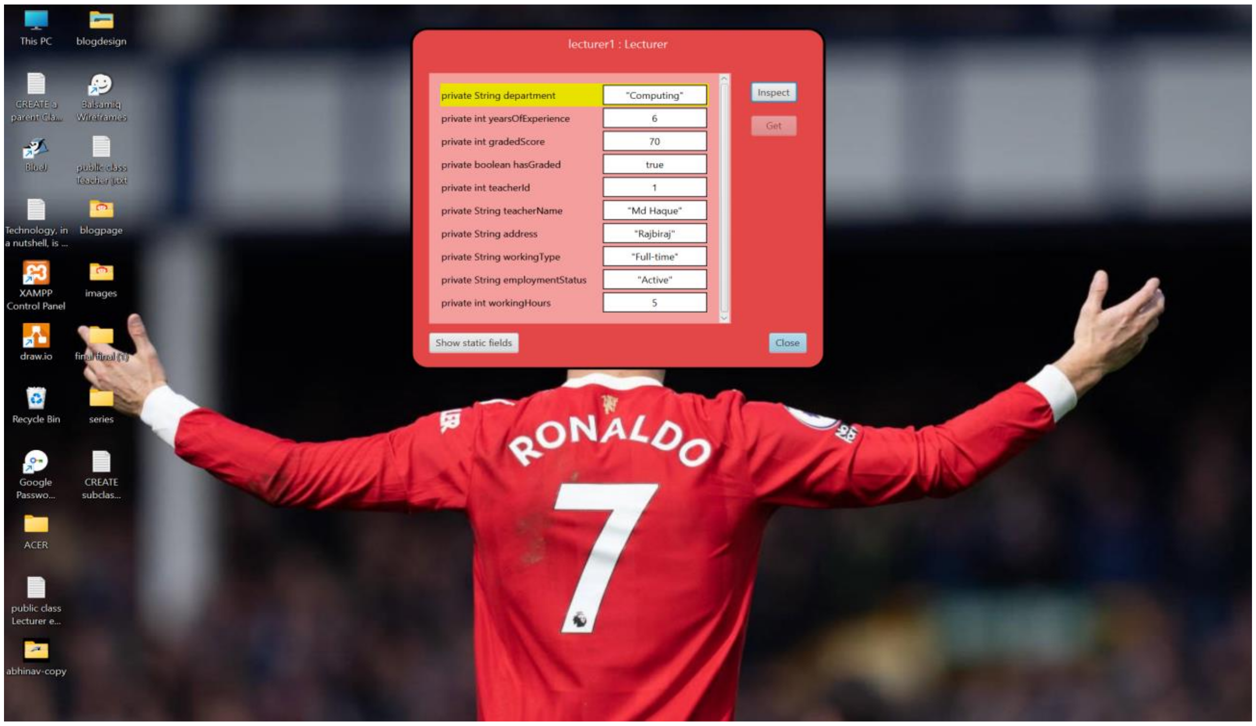
*Table 1: To Inspect the Lecture class, grade the assignment, and re-inspect the Lecturer Class*

## →Output Result:



```
Lecturer(int teacherId, String teacherName, String address, String workingType, int workingHours, String employmentStatus, String department, int yearsOfExperience)

Name of Instance:  lecturer1

new Lecturer( 1 ,
              "Md Haque" ,
              "Rajbiraj" ,
              "Full-time" ,
              5 ,
              "Active" ,
              "Computing" ,
              6 )
```

*Figure 0-1:Screenshot of assigning the data in Lecturer Class*



```
private String department          "Computing"
private int yearsOfExperience      6
private int gradedScore            70
private boolean hasGraded          true
private int teacherId              1
private String teacherName         "Md Haque"
private String address             "Rajbiraj"
private String workingType         "Full-time"
private String employmentStatus    "Active"
private int workingHours           5
```

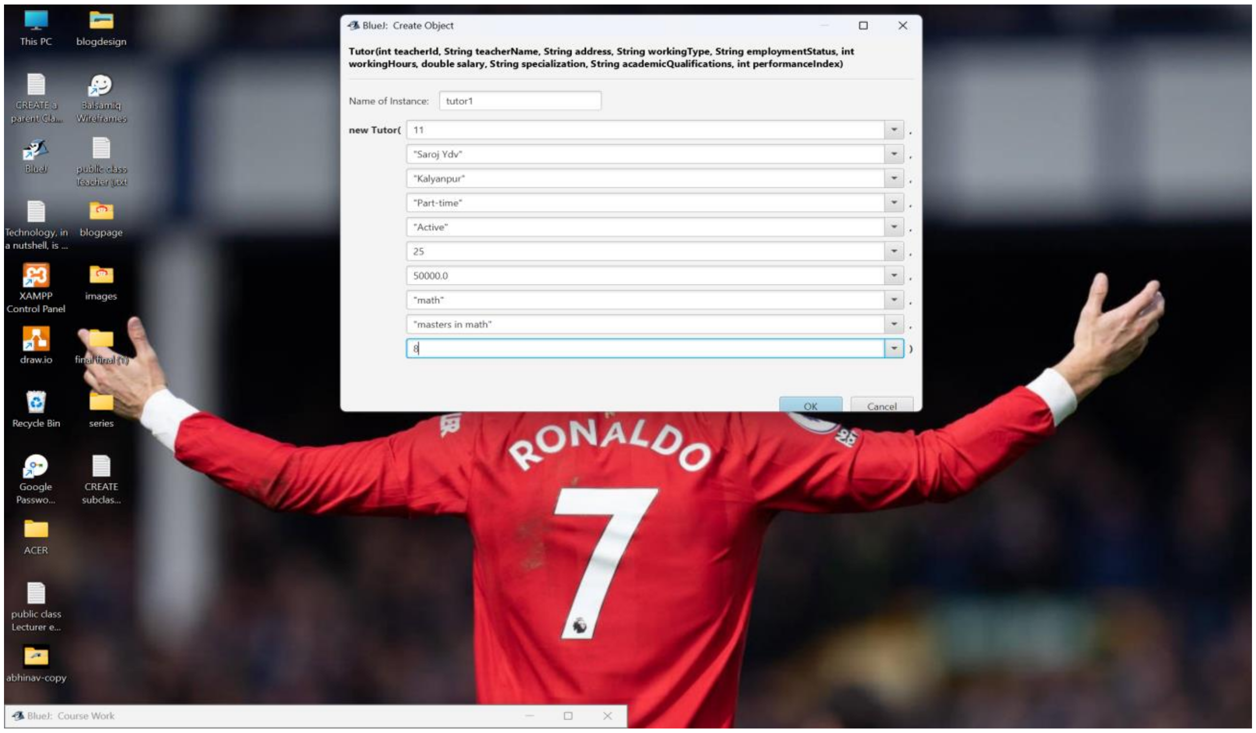
*Figure 0-2:Screenshot for the inspection of Lecturer class*

## 5.2.Test 2 – To inspect Tutor class, set salary and reinspect the Tutor class.

| | |
|---|---|
| Test No: | 2 |
| Objective: | To inspect the Tutor class , set salary and reinspect the Tutor class. |
| Action: | →The Tutor is called with the following arguments:<br>teacherId=11<br>teacherName=”Saroj Ydv”<br>address=”Kalyanpur”<br>workingType=”Part-time”<br>employmentStatus=”Active”<br>workingHours=25<br>salary=50000.0<br>specialization=”math”<br>academicQualifications=”masters in math”<br>perfomanceIndex=8<br>→Inspection of the Tutor class.<br>→void setSalary(double newSalary, int newPerformanceIndex)<br>double newSalary=60000.0<br>int newPerformanceIndex=8<br>→Re-inspection of the Tutor class. |
| Expected Result: | The Tutor would be assign to set the salary. |
| Actual Result: | The salary was set. |
| Conclusion: | The test is successful. |

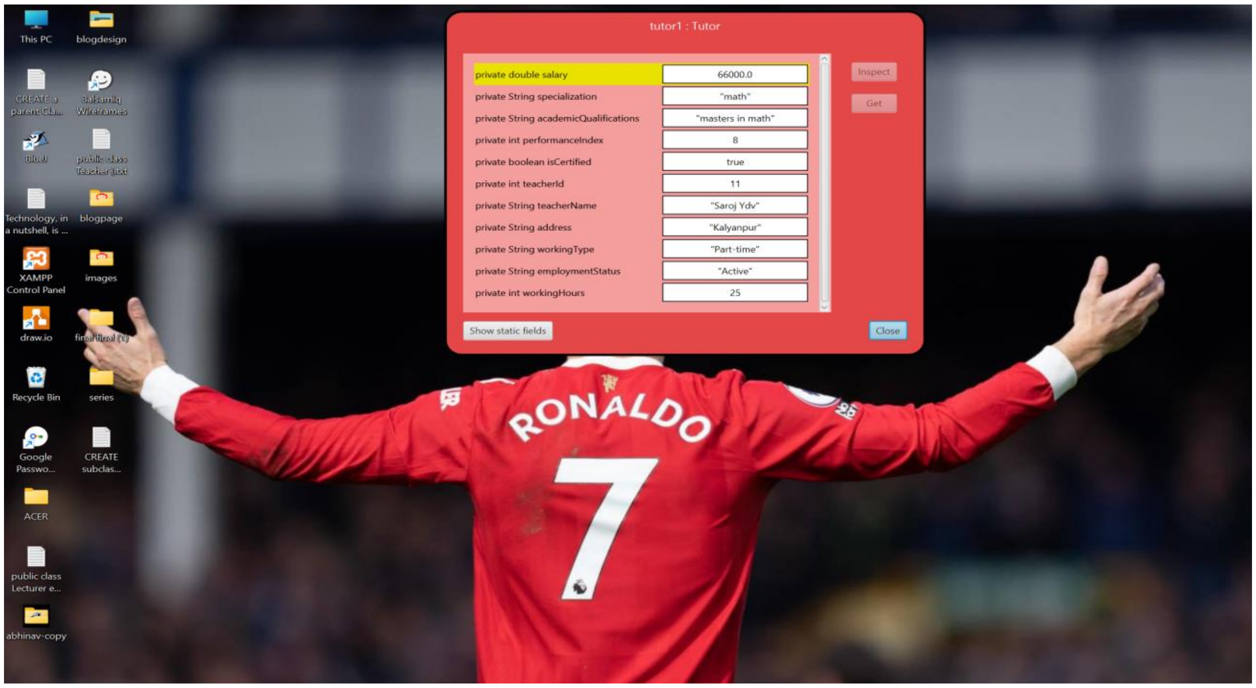
**Table 2: To inspect Tutor class, set salary and reinspect the Tutor class.**

**→Output Result:**



```
BlueJ: Create Object

Tutor(int teacherId, String teacherName, String address, String workingType, String employmentStatus, int
workingHours, double salary, String specialization, String academicQualifications, int performanceIndex)

Name of Instance: tutor1

new Tutor( 11 ,
           "Saroj Ydv" ,
           "Kalyanpur" ,
           "Part-time" ,
           "Active" ,
           25 ,
           50000.0 ,
           "math" ,
           "masters in math" ,
           8 )

OK   Cancel
```

*Figure 0-3:Screeenshot of assigning the data in Tutor class.*



```
tutor1 : Tutor

private double salary                       66000.0
private String specialization               "math"
private String academicQualifications       "masters in math"
private int performanceIndex                8
private boolean isCertified                 true
private int teacherId                       11
private String teacherName                  "Saroj Ydv"
private String address                      "Kalyanpur"
private String workingType                  "Part-time"
private String employmentStatus             "Active"
private int workingHours                    25

Show static fields                          Inspect  Get  Close
```

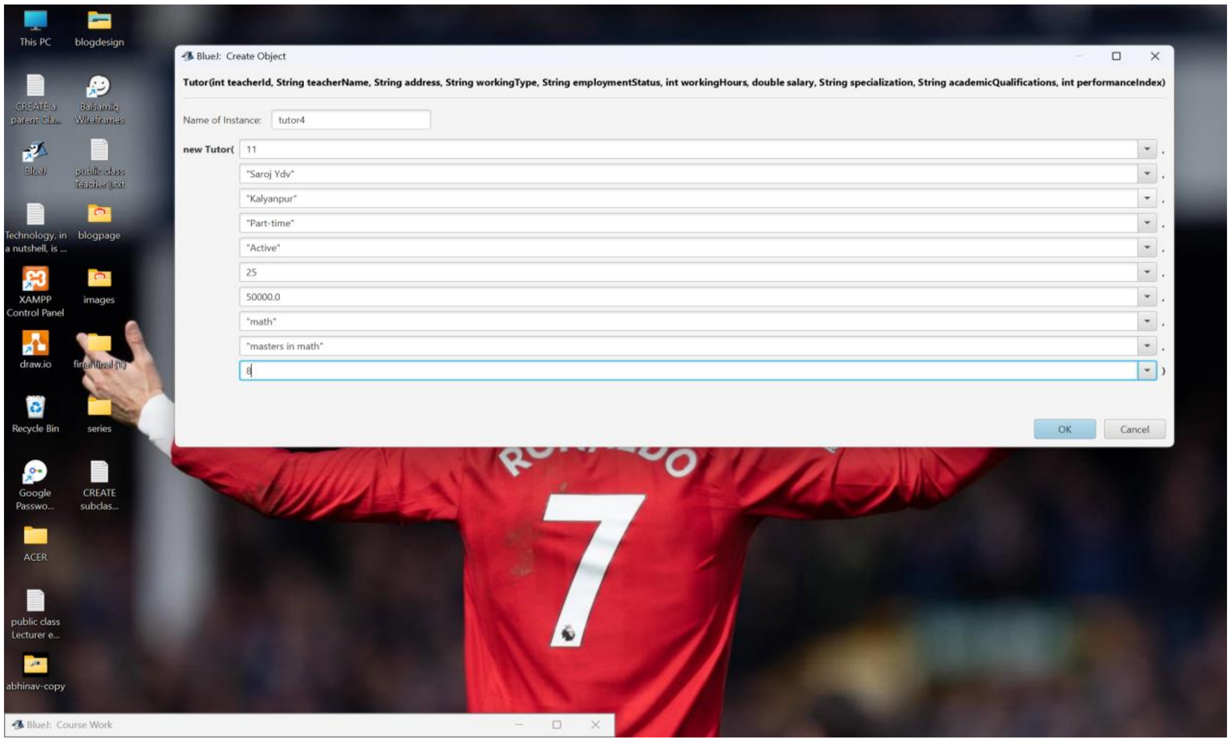
*Figure 0-4Screenshot of inspection of Tutor class*

## 5.3.Test 3: To inspect the Tutor class again after removing the tutor.

| | |
|---|---|
| Test No: | 3 |
| Objective: | To inspect Tutor class again after removing the tutor. |
| Action: | →The Tutor is called with the following arguments:<br>teacherId=11<br>teacherName=”Saroj Ydv”<br>address=”Kalyanpur”<br>workingType=”Part-time”<br>employmentStatus=”Active”<br>workingHours=25<br>salary=50000.0<br>specialization=”math”<br>academicQualifications=”masters in math”<br>perfomanceIndex=8<br>→Inspection of the Tutor class.<br>→void removeTutor()<br>→Re-inspection of the Tutor class |
| Expected Result: | The  uncertified tutor would be removed. |
| Actuall Result: | The uncertified tutor was removed. |
| Conclusion: | The test is successful. |

**Table 3: To inspect the Tutor class again after removing the tutor.**

## →Output Result:



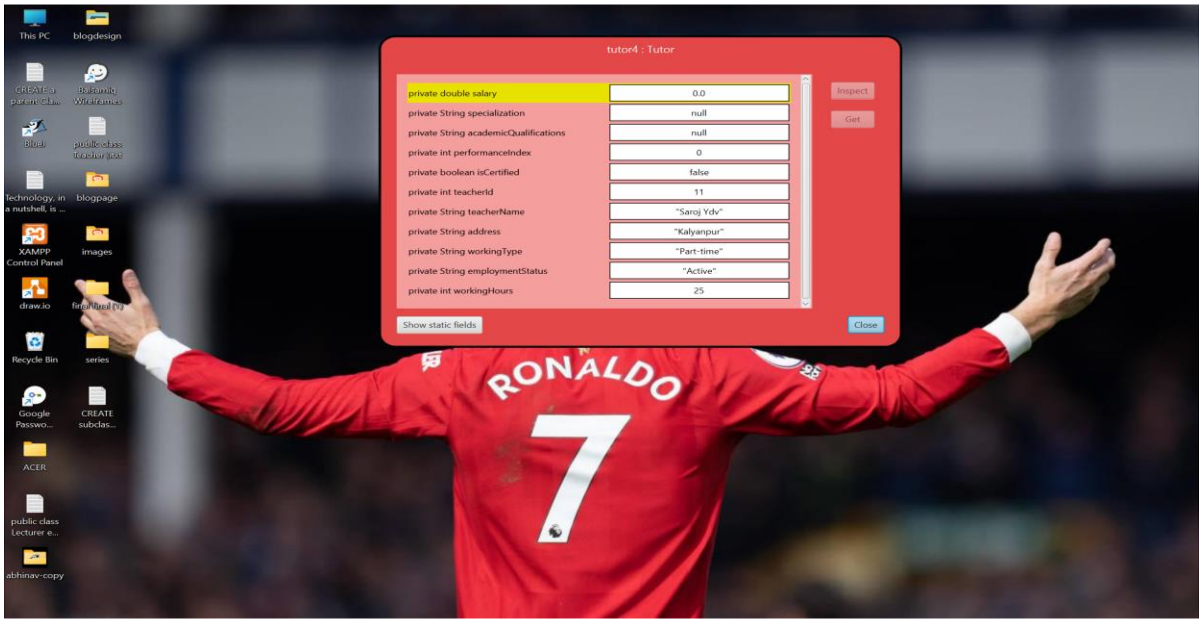
BlueJ: Create Object

Tutor(int teacherId, String teacherName, String address, String workingType, String employmentStatus, int workingHours, double salary, String specialization, String academicQualifications, int performanceIndex)

Name of Instance: tutor4

```
new Tutor( 11 ,
           "Saroj Ydv" ,
           "Kalyanpur" ,
           "Part-time" ,
           "Active" ,
           25 ,
           50000.0 ,
           "math" ,
           "masters in math" ,
           8 )
```

*Figure 0-5Screeenshot of assigning the data in Tutor class.*



tutor4 : Tutor

| | |
|---|---|
| private double salary | 0.0 |
| private String specialization | null |
| private String academicQualifications | null |
| private int performanceIndex | 0 |
| private boolean isCertified | false |
| private int teacherId | 11 |
| private String teacherName | "Saroj Ydv" |
| private String address | "Kalyanpur" |
| private String workingType | "Part-time" |
| private String employmentStatus | "Active" |
| private int workingHours | 25 |

*Figure 0-6:Screenshot of inspection of Tutor class*

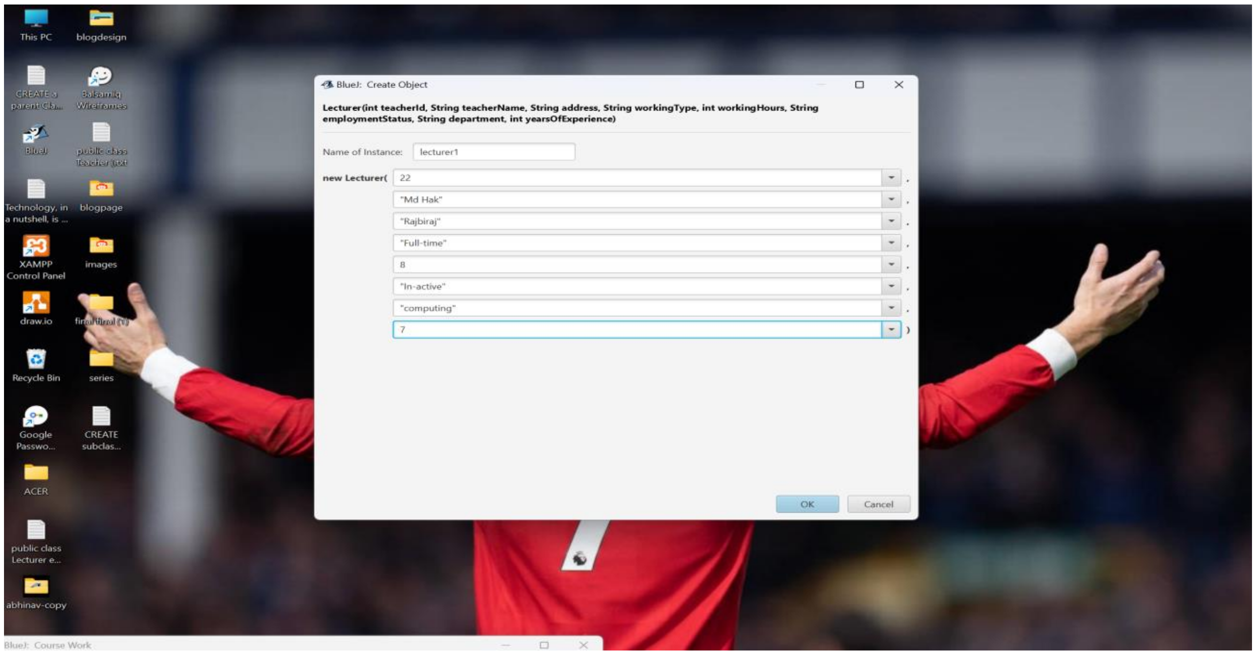
## 5.4.Test 4: To display the details of Lecturer and Tutor classes

| | |
|---|---|
| Test No: | 4 |
| Objective: | To inspect the details of Lecturer and Tutor classes |
| Action: | →The Lecturer class is called with the following arguments:<br>teacherId=22<br>teacherName=”Md Hak”<br>address=”Rajbiraj”<br>workingType=”Full-time”<br>workingHours=8<br>employmentStatus=”In-active”<br>department=”computing”<br>yearsOfExperience=7<br>→Inspection of the Lecturer class.<br>→void displayLecturerDetails:<br>→Re-inspection of the Lecturer class<br><br>→The Tutor class is called with the following arguments:<br>teacherId=22<br>teacherName=”Md Haque”<br>address=”Rajbiraj”<br>workingType=”Part-Time”<br>employmentStatus=”Active”<br>workingHours=21<br>salary=5000.0<br>specialization=”computer”<br>academicQualifications=”masters in computing” |

| | |
|---|---|
| | performanceIndex=8<br>→Inspection of the Tutor class.<br>→void displayTutorDetails:<br>→Re-inspection of the Tutor class |
| Expected Result: | The LecturerDetails would be displayed.<br>The TutorDetails would be displayed. |
| Actual Result: | The LecturerDetails was displayed.<br>The TutorDetails was displayed. |
| Conclusion: | The test is successful. |

**Table 4: To display the details of Lecturer class.**

**→Output Result:**

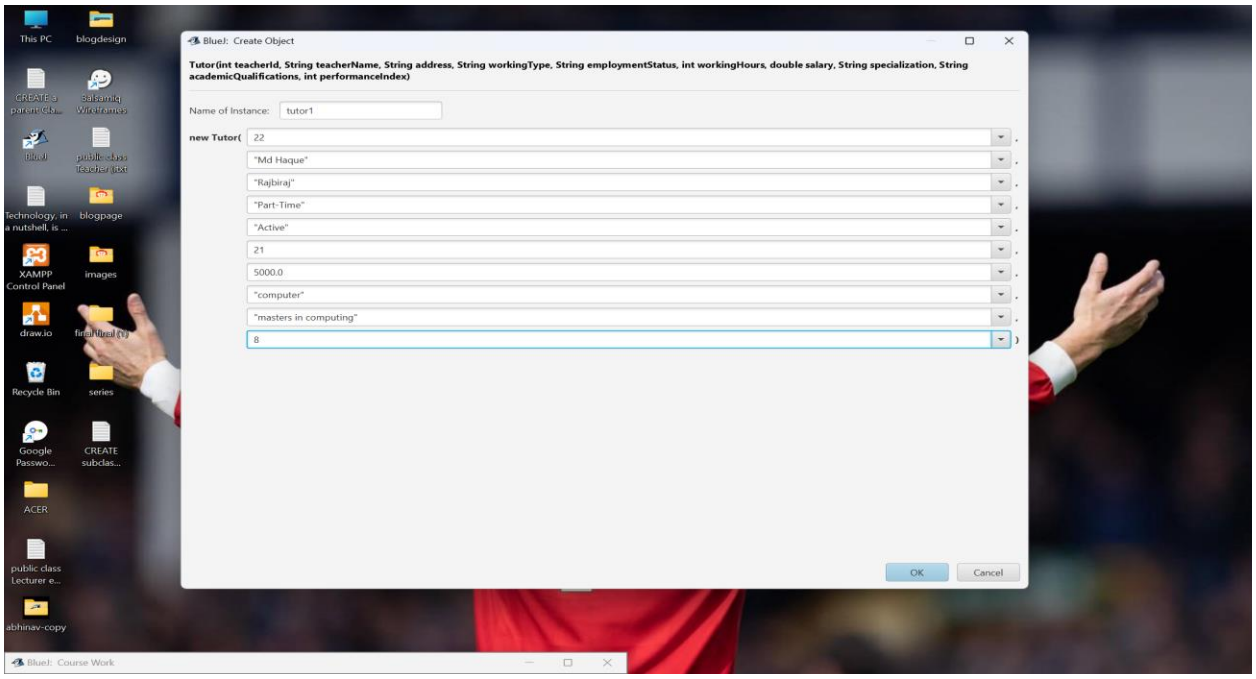

BlueJ: Create Object

**Lecturer(int teacherId, String teacherName, String address, String workingType, int workingHours, String employmentStatus, String department, int yearsOfExperience)**

Name of Instance: lecturer1

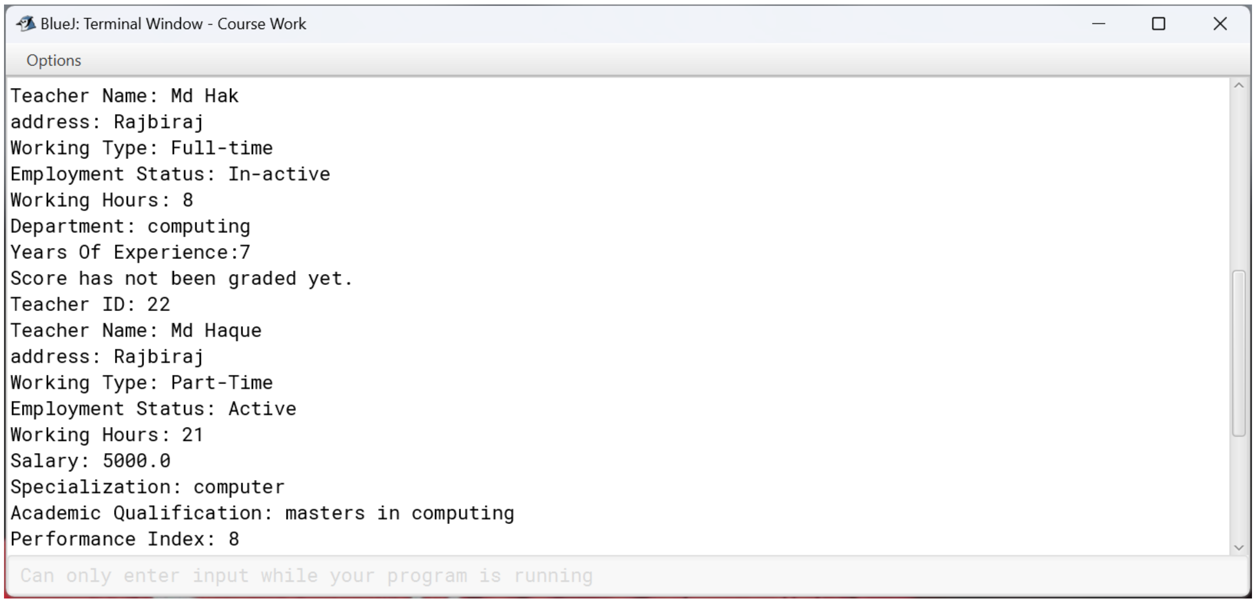
```
new Lecturer( 22 ,
"Md Hak" ,
"Rajbiraj" ,
"Full-time" ,
8 ,
"In-active" ,
"computing" ,
7 )
```

*Figure 0-7:Screeenshot of assigning the data in Lecturer class*



```
Tutor(int teacherId, String teacherName, String address, String workingType, String employmentStatus, int workingHours, double salary, String specialization, String academicQualifications, int performanceIndex)

Name of Instance: tutor1

new Tutor( 22
           "Md Haque"
           "Rajbiraj"
           "Part-Time"
           "Active"
           21
           5000.0
           "computer"
           "masters in computing"
           8 )
```

*Figure 0-8:Screenshot of assigning the data in Tutor class*



```
Teacher Name: Md Hak
address: Rajbiraj
Working Type: Full-time
Employment Status: In-active
Working Hours: 8
Department: computing
Years Of Experience:7
Score has not been graded yet.
Teacher ID: 22
Teacher Name: Md Haque
address: Rajbiraj
Working Type: Part-Time
Employment Status: Active
Working Hours: 21
Salary: 5000.0
Specialization: computer
Academic Qualification: masters in computing
Performance Index: 8
Can only enter input while your program is running
```

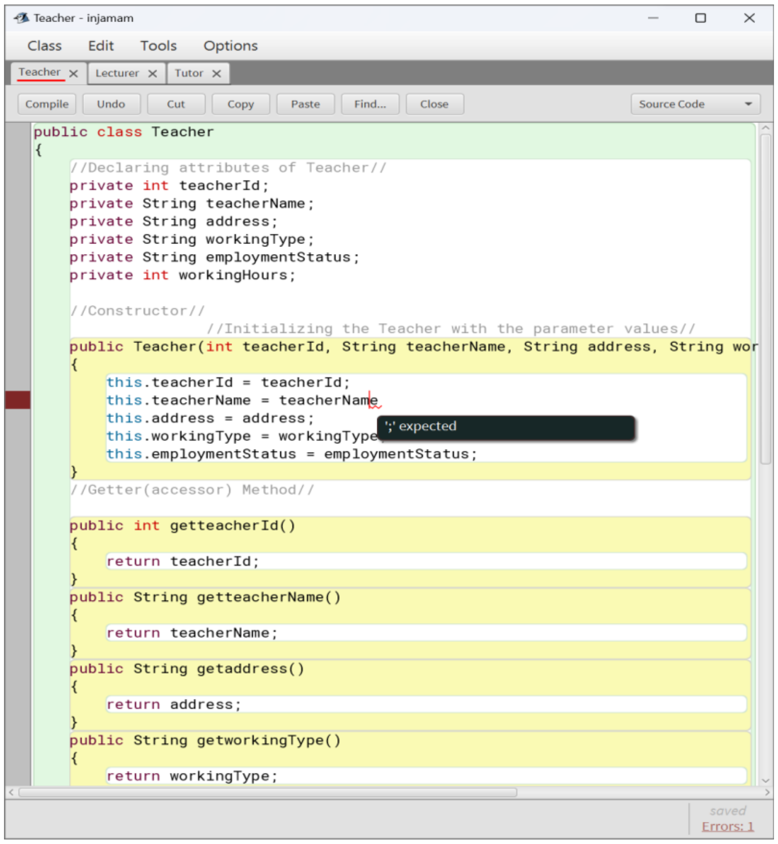
*Figure 0-9:Figure 0 13:Display screenshot of both Lecturer and Tutor classes respectively.*

## 6.Error detection and error correction:

Errors are very common problem in every programming language. Likewise in Java programming language also there are three very common error which can be faced by programmer at any time.There are three types of error in java they are syntax, semantic and logical.

### 6.1.Syntax error:

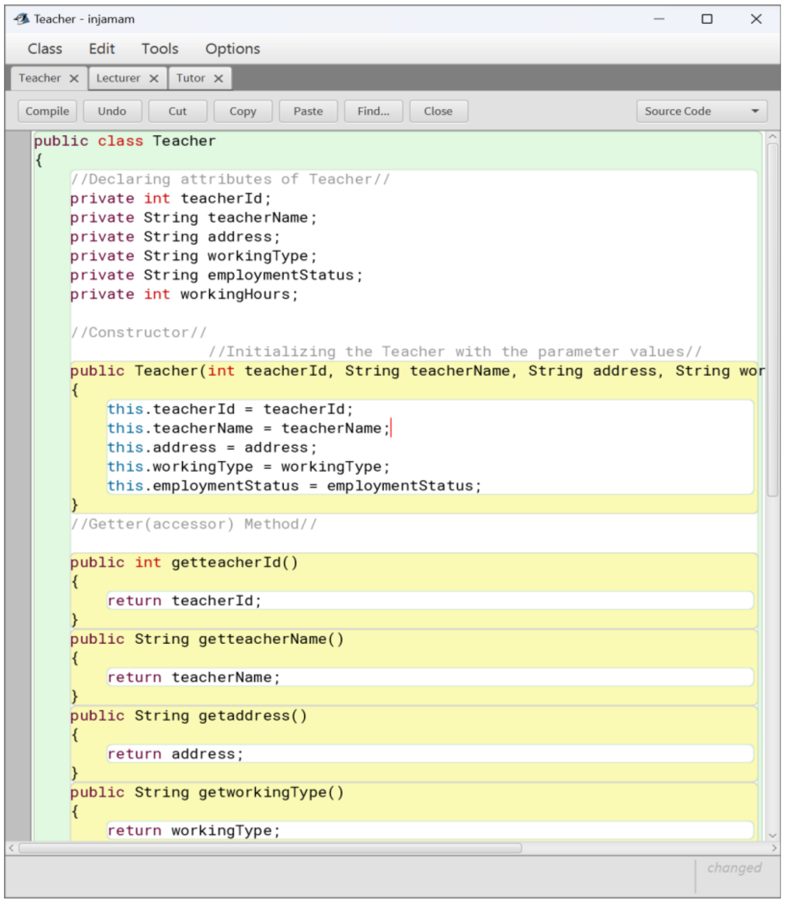
A syntax error is a mistake in the structure of a programming or scripting language, akin to a grammatical error in human language. It prevents code from executing correctly, as the computer cannot understand the intended instructions. (Gaudet, December 27, 2023)



```
public class Teacher
{
    //Declaring attributes of Teacher//
    private int teacherId;
    private String teacherName;
    private String address;
    private String workingType;
    private String employmentStatus;
    private int workingHours;

    //Constructor//
               //Initializing the Teacher with the parameter values//
    public Teacher(int teacherId, String teacherName, String address, String wor
    {
        this.teacherId = teacherId;
        this.teacherName = teacherName
        this.address = address;
        this.workingType = workingType
        this.employmentStatus = employmentStatus;
    }
    //Getter(accessor) Method//

    public int getteacherId()
    {
        return teacherId;
    }
    public String getteacherName()
    {
        return teacherName;
    }
    public String getaddress()
    {
        return address;
    }
    public String getworkingType()
    {
        return workingType;
```

```
';' expected
```

*Figure 0-1:Syntax error before correction*

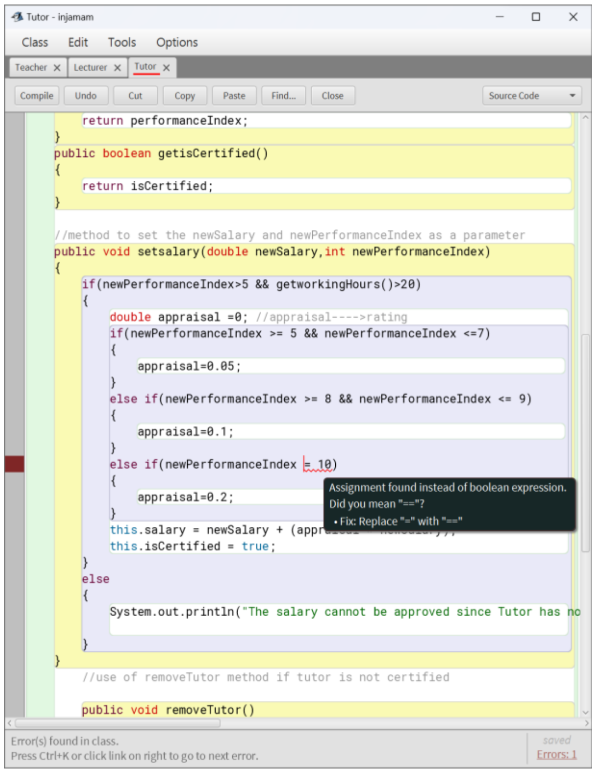


```
public class Teacher
{
    //Declaring attributes of Teacher//
    private int teacherId;
    private String teacherName;
    private String address;
    private String workingType;
    private String employmentStatus;
    private int workingHours;

    //Constructor//
                    //Initializing the Teacher with the parameter values//
    public Teacher(int teacherId, String teacherName, String address, String wor
    {
        this.teacherId = teacherId;
        this.teacherName = teacherName;
        this.address = address;
        this.workingType = workingType;
        this.employmentStatus = employmentStatus;
    }
    //Getter(accessor) Method//

    public int getteacherId()
    {
        return teacherId;
    }
    public String getteacherName()
    {
        return teacherName;
    }
    public String getaddress()
    {
        return address;
    }
    public String getworkingType()
    {
        return workingType;
```

*Figure 0-2:After correction of syntax error*

## 6.2.Semantic error:

The semantic error means If there is a semantic error in your program, it will run successfully in the sense that the computer will not generate any error messages. However, your program will not do the right thing. It will do something else. Specifically, it will do what you told it to do.The problem is that the program you wrote is not the program you wanted to write. The meaning of the program (its semantics) is wrong. Identifying semantic errors can be tricky because it requires you to work backward by looking at the output of the program and trying to figure out what it is doing. ($ingh, August 17, 2018)



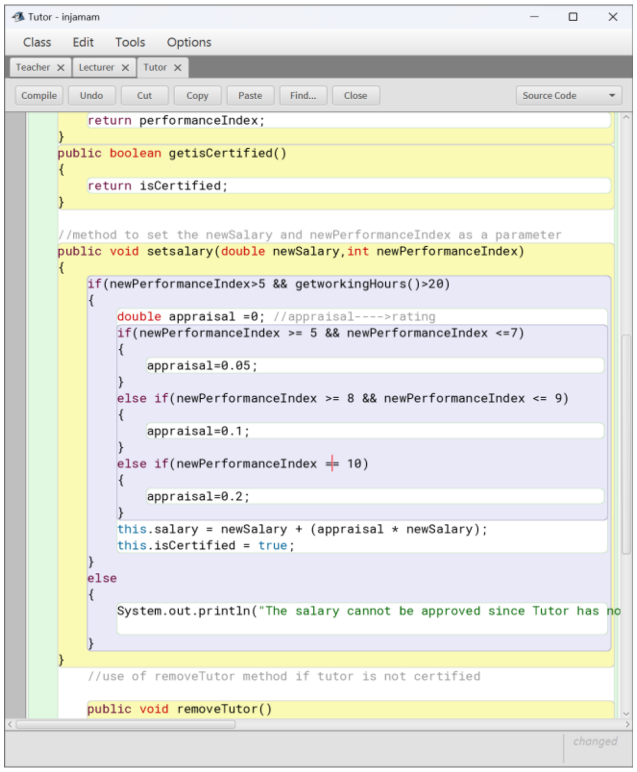
```
        return performanceIndex;
    }
    public boolean getisCertified()
    {
        return isCertified;
    }

    //method to set the newSalary and newPerformanceIndex as a parameter
    public void setsalary(double newSalary,int newPerformanceIndex)
    {
        if(newPerformanceIndex>5 && getworkingHours()>20)
        {
            double appraisal =0; //appraisal---->rating
            if(newPerformanceIndex >= 5 && newPerformanceIndex <=7)
            {
                appraisal=0.05;
            }
            else if(newPerformanceIndex >= 8 && newPerformanceIndex <= 9)
            {
                appraisal=0.1;
            }
            else if(newPerformanceIndex = 10)
            {
                appraisal=0.2;
            }
            this.salary = newSalary + (appr
            this.isCertified = true;
        }
        else
        {
            System.out.println("The salary cannot be approved since Tutor has no

        }
    }
        //use of removeTutor method if tutor is not certified

        public void removeTutor()
```

Assignment found instead of boolean expression.
Did you mean "=="?
• Fix: Replace "=" with "=="

Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.

saved
Errors: 1

*Figure 0-3:Sementic error before correction*



```
        return performanceIndex;
    }
    public boolean getisCertified()
    {
        return isCertified;
    }

    //method to set the newSalary and newPerformanceIndex as a parameter
    public void setsalary(double newSalary,int newPerformanceIndex)
    {
        if(newPerformanceIndex>5 && getworkingHours()>20)
        {
            double appraisal =0; //appraisal---->rating
            if(newPerformanceIndex >= 5 && newPerformanceIndex <=7)
            {
                appraisal=0.05;
            }
            else if(newPerformanceIndex >= 8 && newPerformanceIndex <= 9)
            {
                appraisal=0.1;
            }
            else if(newPerformanceIndex == 10)
            {
                appraisal=0.2;
            }
            this.salary = newSalary + (appraisal * newSalary);
            this.isCertified = true;
        }
        else
        {
            System.out.println("The salary cannot be approved since Tutor has no

        }
    }
        //use of removeTutor method if tutor is not certified

        public void removeTutor()
```

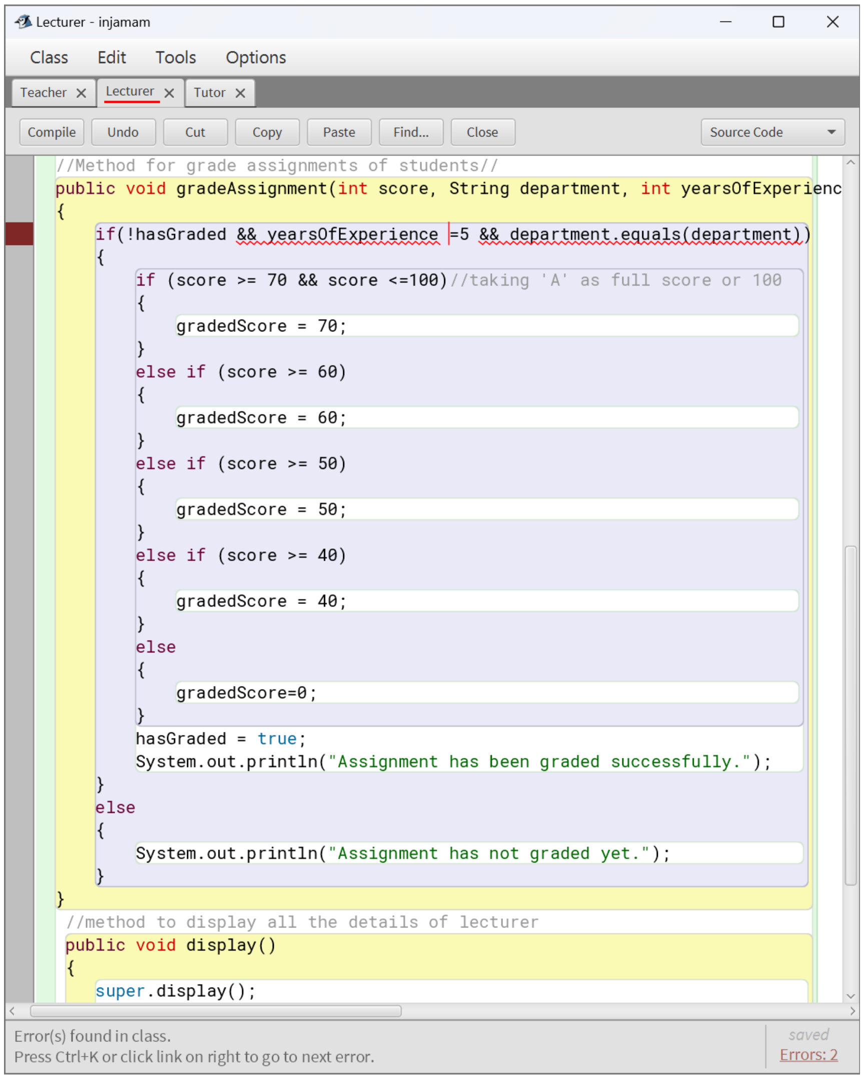
changed

*Figure 0-4:After correction of sementic error*

## 6.3.Logical error:

These occur when the program is executing correctly, but the result is not what was intended. These errors can be difficult to identify and fix, as the program is running correctly but producing unintended results.

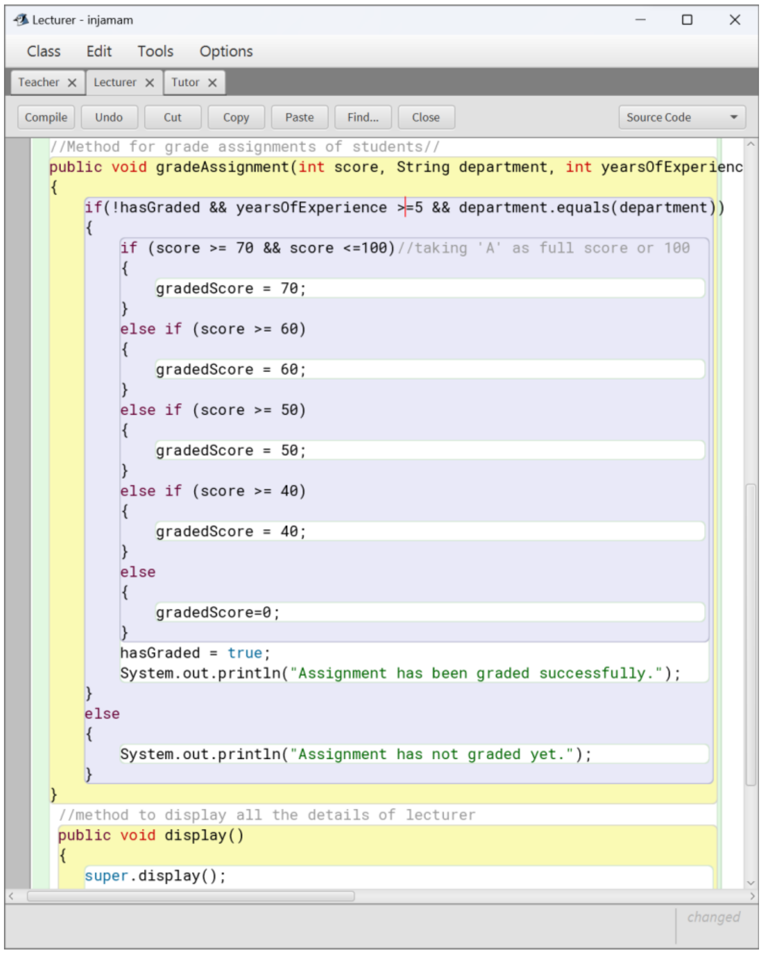
(Vyas, September 9,2023)



```
//Method for grade assignments of students//
public void gradeAssignment(int score, String department, int yearsOfExperienc
{
    if(!hasGraded && yearsOfExperience =5 && department.equals(department))
    {
        if (score >= 70 && score <=100)//taking 'A' as full score or 100
        {
            gradedScore = 70;
        }
        else if (score >= 60)
        {
            gradedScore = 60;
        }
        else if (score >= 50)
        {
            gradedScore = 50;
        }
        else if (score >= 40)
        {
            gradedScore = 40;
        }
        else
        {
            gradedScore=0;
        }
        hasGraded = true;
        System.out.println("Assignment has been graded successfully.");
    }
    else
    {
        System.out.println("Assignment has not graded yet.");
    }
}
 //method to display all the details of lecturer
 public void display()
 {
    super.display();
```

```
Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.
```

*Figure 0-5:Logical error before correction*



```
//Method for grade assignments of students//
public void gradeAssignment(int score, String department, int yearsOfExperienc
{
    if(!hasGraded && yearsOfExperience >=5 && department.equals(department))
    {
        if (score >= 70 && score <=100)//taking 'A' as full score or 100
        {
            gradedScore = 70;
        }
        else if (score >= 60)
        {
            gradedScore = 60;
        }
        else if (score >= 50)
        {
            gradedScore = 50;
        }
        else if (score >= 40)
        {
            gradedScore = 40;
        }
        else
        {
            gradedScore=0;
        }
        hasGraded = true;
        System.out.println("Assignment has been graded successfully.");
    }
    else
    {
        System.out.println("Assignment has not graded yet.");
    }
}
 //method to display all the details of lecturer
 public void display()
 {
    super.display();
```

*Figure 0-6:After correction of logical error*

## 7.Conclusion:

This assignment has provided a practical application of object-oriented programming concepts, allowing me to gain a deeper understanding of class hierarchies, inheritance, encapsulation, and method overriding. By implementing the assignment, my proficiency in Java programming has been enhanced, enabling me to design and structure classes based on real-world scenarios more effectively.I have learned to class diagram, Pseudocode and also to use various software tools like Ms-word, Draw.io and so on. Overall to be seen this coursework helped me to tackle various real world things also helped me to learn broad .

I successfully implemented three classes: Teacher, Lecturer, and Tutor, establishing a hierarchical structure to represent various roles within an educational system. Each class possesses appropriate attributes, constructors, accessor methods, mutator methods, and display methods.

The Teacher class demonstrates adequate encapsulation, proper utilization of getter and setter methods, and a clear display method.The Lecturer class extends the Teacher class, additional attributes, and includes a method (gradeAssignment) for grading student assignments.The Tutor class inherits from the Teacher class,which includes a method (setSalary) for salary calculation, and effectively handles the removal of tutors or remove tutor method.The setSalary method is used in this class.Ensuring the accurate implementation of conditions for grading assignments required a huge research and findings.Validation of input in methods such as setWorkingHours and setSalary, grading assignment proved to be challenging. I resorted to online resources and documentation for guidance. Seeking advice from programming communities and forums proved to be immensely beneficial as it offered valuable insights.

Overall this assignment helped me to cover approximately all things which I learned in first semester.This task has really presented a valuable opportunity to apply theoretical knowledge of object-oriented programming in a practical setting. While the implementation of the concepts, there is always room for enhancement. Consistently refining the code, addressing logical complexities. Also, it has developed a good habit of research and findings. There were various problems which I was facing while making the coursework but afterall difficulties I was able to solve the realworld scenarios. However, my knowledge on OOP gradually increased and after this I am confident to solve these types of problems.I have learned various methods. At first I was facing alot of difficulties slowly and gradually I was able to cover my coursework.

The experience gained from this assignment has further solidified my comprehension of Java.

## 8. References

$ingh, M., August 17, 2018. *Sololearn.* [Online]
Available at: https://www.sololearn.com/en/Discuss/1458964/what-is-semantic-error
[Accessed 17 8 2018].

Ali, S. W., September 2018. *ResearchGate.* [Online]
Available at:
https://www.researchgate.net/publication/329140153_Blue_J_Programming
[Accessed 9 2018].

Anon., n.d. [Online].

Baird, K., Ovtober 21, 2021. *ellipsis.* [Online]
Available at: https://ellipsiseducation.com/blog/what-is-java
[Accessed 21 10 2021].

Ballew, J., June 12,2021. *LifeWire.* [Online]
Available at: https://www.lifewire.com/microsoft-word-4159373
[Accessed 12 6 2021].

Gaudet, H., December 27, 2023. *EasyThinkJunkie.* [Online]
Available at: https://www.easytechjunkie.com/what-is-a-syntax-error.htm
[Accessed 27 12 2023].

Mae, A., November 18, 2022. *aiseesoft.* [Online]
Available at: https://www.nuclino.com/apps/drawio
[Accessed 18 11 2022].

Obvii, October 7 2022. *noon.* [Online]
Available at: https://www.learnatnoon.com/s/en-pk2/definition-of-pseudo-code/
[Accessed 7 10 2022].

Pedamkar, P., March 13,2023. *EDUCBA.* [Online]
Available at: https://www.educba.com/class-diagram/
[Accessed 13 3 2023].

Vyas, H., September 9,2023. *codingninjas.* [Online]
Available at: https://www.codingninjas.com/studio/library/types-of-error-in-java
[Accessed 9 9 2023].

## 9.Appendix

### 9.1.Code Of Parent class Teacher:

```
public class Teacher

{

    //Declaring attributes of Teacher//

    private int teacherId;

    private String teacherName;

    private String address;

    private String workingType;

    private String employmentStatus;

    private int workingHours;
    //Constructor//
                    //Initializing the Teacher with the parameter values//
    public Teacher(int teacherId, String teacherName, String address, String
workingType, String employmentStatus)
    {
        this.teacherId = teacherId;
        this.teacherName = teacherName;
        this.address = address;
        this.workingType = workingType;
        this.employmentStatus = employmentStatus;
    }
    //Getter(accessor) Method//

    public int getteacherId()
    {
        return teacherId;
    }
    public String getteacherName()
    {
        return teacherName;
    }
    public String getaddress()
    {
        return address;
    }
    public String getworkingType()
    {
        return workingType;
    }
    public String getemploymentStatus()
    {
        return employmentStatus;
    }
     public int getworkingHours()
    {
        return workingHours;
    }

    //Setter Method for Teacher WorkingHours//
    public void setworkingHours(int workingHours)
    {
        this.workingHours = workingHours;
    }

    //Display Method//
    public void displayTeacherDetails()
    {
        System.out.println("Teacher ID: "+getteacherId());
        System.out.println("Teacher Name: "+getteacherName());
        System.out.println("address: "+getaddress());
        System.out.println("Working Type: "+getworkingType());
        System.out.println("Employment Status: "+getemploymentStatus());
        if(workingHours <= 0)
        {
           System.out.println("Working Hours: Not assigned ");
        }
        else
        {
             System.out.println("Working Hours: " +workingHours);
        }

    }
}
```

## 9.2.Code Of Sub-class Lecturer:

```
public class Lecturer extends Teacher
   {
   //additional attributes for subclass lecturer
   private String department;
   private int yearsOfExperience;
   private int gradedScore;
   private boolean hasGraded;
   //constructors//
   public Lecturer(int teacherId, String teacherName, String address, String
workingType, int workingHours, String employmentStatus, String department, int
yearsOfExperience)
   {
      //super class constructors
      super(teacherId, teacherName, address, workingType, employmentStatus);
      //call for WorkingHours
      super.setworkingHours(workingHours);



      this.department = department;
```

```
        this.yearsOfExperience = yearsOfExperience;

        this.gradedScore = 0;

        this.hasGraded = false;

    }

    //  getter method(accessor) //

    public String getdepartment()

    {

        return department;

    }

    public int getyearsOfExperience()

    {

        return yearsOfExperience;

    }

    public int getgradedScore()

    {

        return gradedScore;

    }

    public void setgradedScore(int gradedScore)//setter method for gradedScore

    {

        this.gradedScore = gradedScore;

    }

    public boolean gethasGraded()

    {
        return hasGraded;
    }
    //Method for grade assignments of students//
    public void gradeAssignment(int score, String department, int yearsOfExperience)
    {
        if(!hasGraded && yearsOfExperience >=5 && department.equals(department))
        {
            if (score >= 70 && score <=100)//taking 'A' as full score or 100
            {
                gradedScore = 70;
            }
            else if (score >= 60)
            {
                gradedScore = 60;
            }
            else if (score >= 50)
            {
                gradedScore = 50;
            }
            else if (score >= 40)
            {
                gradedScore = 40;
            }
            else
            {
                gradedScore=0;
            }
            hasGraded = true;
            this.gradedScore=score;
            System.out.println("Assignment has been graded successfully");
        }
        else
        {
            System.out.println("Assignment hasn't been graded yet ");
        }
    }
        //@Override
     public void displayLecturerDetails()
     {
        super.displayTeacherDetails();
        System.out.println("Department: "+getdepartment());
        System.out.println("Years Of Experience:" +getyearsOfExperience());
        if(hasGraded)
        {
            System.out.println("Graded Score: "+getgradedScore());
        }
        else

        {

            System.out.println("Score has not been graded yet");

        }

     }

    }
```

## 9.3.Code Of sub-class Tutor:

```
class Tutor extends Teacher
{
    //
    private double salary;
    private String specialization;
    private String academicQualifications;
    private int performanceIndex;
    private boolean isCertified;

    //constructors
    public Tutor(int teacherId, String teacherName, String address, String workingType,
String employmentStatus, int workingHours, double salary,String specialization, String
academicQualifications, int performanceIndex)
    {
        super(teacherId, teacherName, address, workingType,employmentStatus);
        super.setworkingHours(workingHours);//setter method for workingHours

        this.salary=salary;
        this.specialization=specialization;
        this.academicQualifications=academicQualifications;
        this.performanceIndex=performanceIndex;
        this.isCertified=false;
    }
    //accessor(getter) method
    public double getsalary()
    {
        return salary;
    }
    public String getspecialization()
    {
        return specialization;
    }
    public String getacademicQualifications()
    {
        return academicQualifications;
    }
    public int getperformanceIndex()
    {
        return performanceIndex;
    }
    public boolean isCertified()
    {
        return isCertified;
    }
```

```
//method to set the newSalary and newPerformanceIndex as a parameter
public void setSalary(double newSalary,int newPerformanceIndex)
{
    if(newPerformanceIndex>5 && getworkingHours()>20)
    {
        double appraisal =0.0; //appraisal---->rating //
        if(newPerformanceIndex >= 5 && newPerformanceIndex <=7)
        {
            appraisal=0.05;
        }
        else if(newPerformanceIndex >= 8 && newPerformanceIndex <= 9)
        {
            appraisal=0.1;
        }
        else if(newPerformanceIndex == 10)
        {
            appraisal=0.2;
        }

        this.salary = newSalary + (appraisal * newSalary);
        this.isCertified = true;
    }
```

```
        else
        {
            System.out.println("The salary cannot be approved since Tutor has not been
certified");


        }


    }
        //use of removeTutor method if tutor is not certified
        //setter method
        public void removeTutor(){
        if (!isCertified){
            this.salary=0.0;
            this.specialization=null;
            this.academicQualifications=null;
            this.performanceIndex=0;
            isCertified=false;
        }
        else
        {
            System.out.println("Certified Tutor cannot be removed");
        }
        }
        //dispaly method
        public void displayTutorDetails()
        {
            super.displayTeacherDetails();
            if(!isCertified)   // to dispaly all the details of Tutor
            {
                System.out.println("Salary: "+getsalary());
                System.out.println("Specialization: "+getspecialization());
                System.out.println("Academic Qualification: "+getacademicQualifications());
                System.out.println("Performance Index: "+getperformanceIndex());
            }
            else
            {
                System.out.println("The Tutor has not been certified yet.");
            }
        }

    }
```